

Exhibit 36
Filed Under Seal

Contains Highly Confidential AEO and Source Code Materials

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

GOOGLE LLC,

Plaintiff,

v.

SONOS, INC.,

Defendant.

CASE NO. 3:20-cv-06754-WHA

Related to CASE NO. 3:21-cv-07559-WHA

**OPENING EXPERT REPORT OF SAMRAT BHATTACHARJEE REGARDING
INVALIDITY OF U.S. PATENT NOS. 10,779,033 AND 9,967,615 AND OTHER ISSUES**

HIGHLY CONFIDENTIAL AEO AND SOURCE CODE MATERIALS

Contains Highly Confidential AEO and Source Code Materials

could not be prior art to the current application. On September 13, 2019, the USPTO issued a Final Rejection, rejecting the independent claims over Imai (US 2007/0053514) in view of Roberts (US 2012/0304233).

73. When the initial '237 application was filed on December 30, 2011 the applicant included 20 claims. These claims were directed at a multimedia playback device that could "pass information regarding multimedia content to [a] device on the local playback network in response to a trigger." The independent Claims 1, 10 and 20 are reproduced below:

1. A method to provide content to a local playback network, the method comprising:
 identifying multimedia content from a content provider;
 passing information regarding the multimedia content to a local playback system including one or more multimedia playback devices in response to a trigger; and
 facilitating play of the multimedia content via a local playback network associated with the local playback system.

10. A computer readable storage medium including instructions for execution by a processor, the instructions, when executed, cause the processor to implement a method to provide content to a local playback network, the method comprising:
 identifying multimedia content from a content provider;
 passing information regarding the multimedia content to a local playback system including one or more multimedia playback devices in response to a trigger; and
 facilitating play of the multimedia content via a local playback network associated with the local playback system.

Contains Highly Confidential AEO and Source Code Materials

20. A multimedia playback device comprising:

- a wireless communication interface to communicate with a local playback network and a multimedia content source;
- a processor to:
 - identify multimedia content from the multimedia content source;
 - pass information regarding the multimedia content to device on the local playback network in response to a trigger; and
 - facilitate play of the multimedia content via the devices on the local playback network.

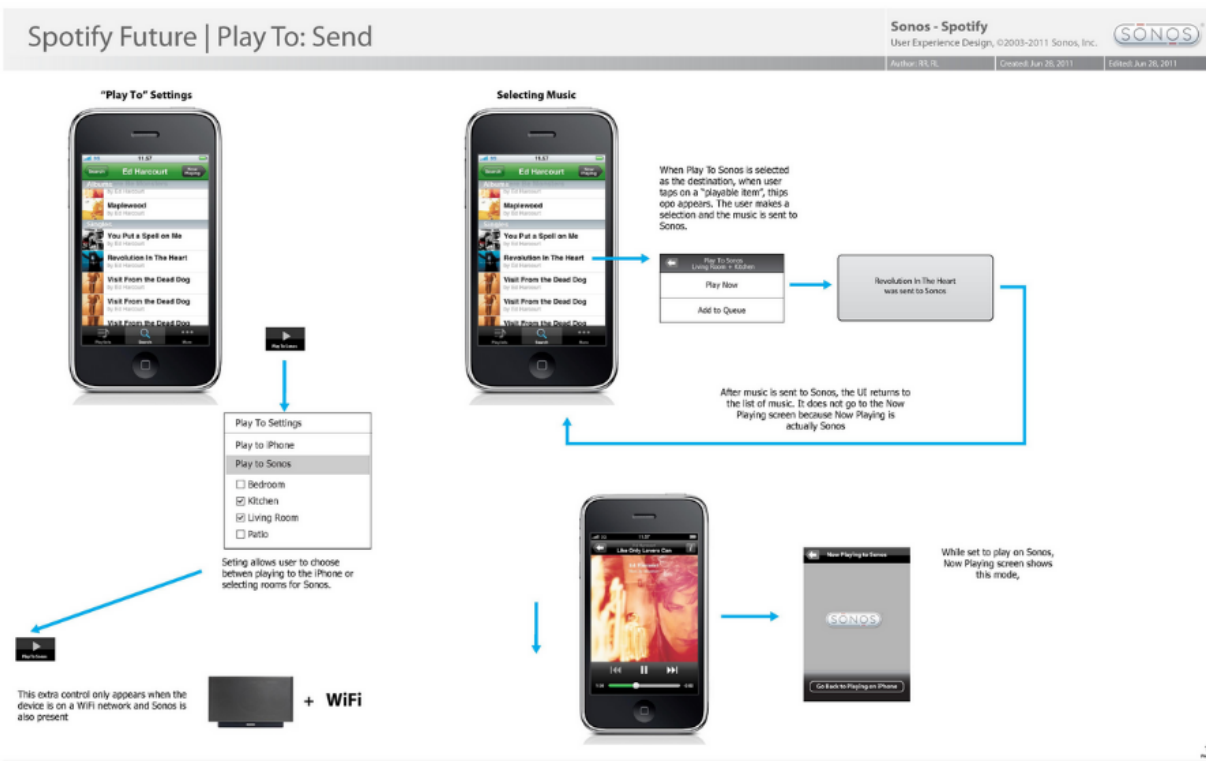
74. On April 25, 2013, the patentee amended the claims as shown below to further clarify that an identification of the multimedia content and the user access rights are passed from the controller to a local playback system, which then enables the local playback system to retrieve the multimedia content from the content provider for playback. Independent Claim 1 is representative of the amendment and reproduced below:

1. (Currently Amended) A method to provide content to a local playback network, the method comprising:

- identifying an item on a controller based on user input, wherein multimedia content associated with the item is retrievable from a content provider with a user access right on the controller; and
- passing information regarding the multimedia content from the controller to a local playback system including one or more multimedia playback devices in response to a trigger, wherein the information includes an identification of the multimedia content and the user access right, and wherein the information enables the local playback system to independently retrieve the multimedia content from the content provider for playback by; and
- ~~facilitating play of the multimedia content via a local playback network associated with the local playback system.~~

75. The applicant made substantial amendments to the claims by the time it filed the '033 patent on April 19, 2019, including to add numerous new limitations regarding: (i) "operating

Contains Highly Confidential AEO and Source Code Materials



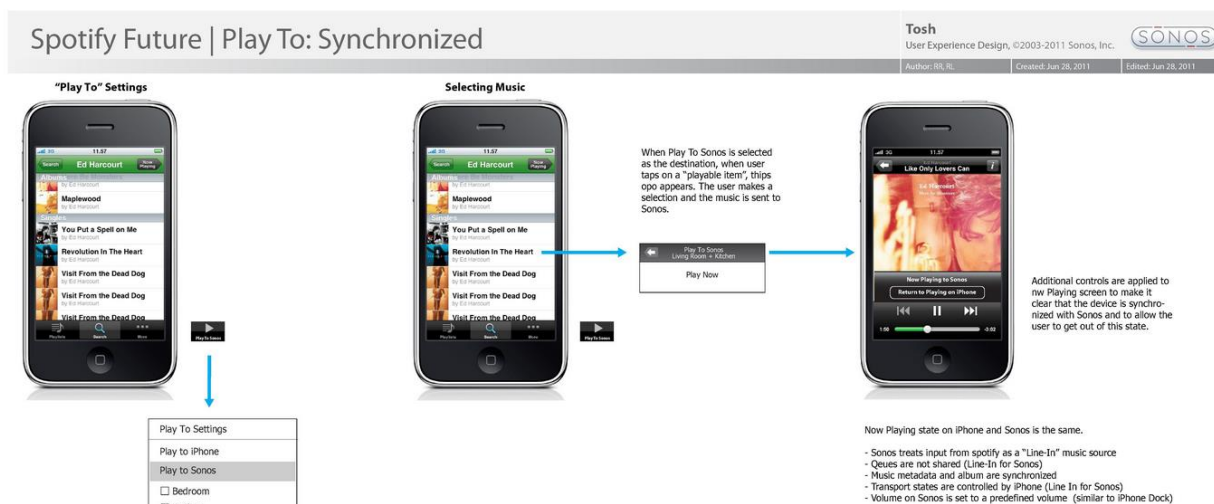
97. First, the wireframe shows that a user selects a "Play to Sonos" icon in a third-party application (Spotify) and then select rooms containing Sonos speakers. For example, in the wireframe the user has selected "Kitchen" and "Living Room".

98. Second, the user taps on a "playable item" (e.g., a music track) he or she would like to play on the Sonos speaker. For instance, in the wireframe the user has selected the song "Revolution In The Heart by Ed Harcourt." Tapping on a playable item presents the user with two options "Play Now" and "Add to Queue." In both options, the wireframe indicates that the playable item (e.g., an identifier for Revolution in The Heart) is "sent to Sonos." The wireframe refers to the Sonos speaker as "Sonos"—e.g., on the bottom left of the above image a Sonos speaker is shown and there is text explaining that the Play to Sonos button appears where "Sonos is also present." Thus, a POSITA would understand this wireframe to, at best, disclose that a user may either (1) select Play Now to send the playable item to the Sonos speaker without being

Contains Highly Confidential AEO and Source Code Materials

added to the local playback system's queue, or (2) select Add to Queue to add the playable item to the queue on the local playback system. There is no disclosure of any cloud queue.

99. The image below reproduces the wireframe for the "Line-in model." The Line-in model does not include the ability to add a playback item to a Sonos queue. Instead, a user may only select a playback item to "Play Now" on the Sonos speaker. Indeed, the wireframe explains "Queues are not shared" in the Line-In mode. In other words, the third-party application (Spotify) has a queue and the local playback system (Sonos) has its separate queue.



100. The wireframes that Mr. Lambourne attached to his July 18, 2011 email do not disclose any cloud queue and Mr. Lambourne's emails and wireframe attachment fail to disclose that Sonos was in possession of an invention containing all of the following limitations, including:

- operating in a first mode in which the computing device is configured for playback of a remote playback queue provided by a cloud-based computing system associated with a cloud-based media service;
- transmitting an instruction for the at least one given playback device to take over responsibility for playback of the remote playback queue from the computing

Contains Highly Confidential AEO and Source Code Materials

understand Mr. Kuper, Mr. Singh, and Mr. Lambourne are not inventors of the '033 patent and thus could not have conceived of the claimed invention.

104. In any event, neither of these documents demonstrate conception of the claimed invention. I have reproduced the July 15 email from Ron Kuper below:

To: Ron Kuper[Ron.Kuper@sonos.com]; Rob Lambourne[Rob.Lambourne@sonos.com]; Joni Hoadley[Joni.Hoadley@sonos.com]; Robert Reimann[Robert.Reimann@sonos.com]
Cc: Andrew Schuler[Andrew.Schuler@sonos.com]; David Taylor[David.Taylor@sonos.com]; Jerry Anderson[Jerry.Anderson@sonos.com]; John Rodley[John.Rodley@sonos.com]; Nick Millington[Nick.Millington@sonos.com]
From: Tad Coburn[/O=MSTEXCHANGE/OU=SOS ADMIN GROUP/CN=RECIPIENTS/CN=TAD.COBURN]
Sent: Fri 7/15/2011 10:16:14 AM Eastern Daylight Time
Subject: FW: Play To Sonos
Attachment: PlayToSonos.pdf

John Rodley has asked to join the conversation as well, so I'm c'ing him.

From: Ron Kuper
Sent: Friday, July 15, 2011 10:10 AM
To: Tad Coburn; Andrew Schuler; Rob Lambourne; Joni Hoadley; Robert Reimann; Nick Millington; Jerry Anderson; David Taylor
Subject: RE: Play To Sonos

Tad asked me to diagram how the play-to-Sonos would work using the IDs thrown over into the cloud. I've made a first pass at this in the picture.

One piece of this is that we want to introduce a new centralized web service for pushing events out to Sonos equipment. The idea is that any Zone or CR that wants to get just-in-time events connects to this service, and then asks to listen for events that match a particular pattern. We would use this service not just for Play-to Sonos, but also to allow us to get just-in-time updates to playlists, starred tracks, etc. (I also envision using this service for any time we'd want to push events out to our equipment – maybe someday we'd want CS to be able to poke at customer's system if necessary, or we could hook this into our upgrade mechanism somehow.)

Assuming we have this event service, I am envisioning an event called PlayFrom, that takes as parameters (conceptually) the following:

- Service info – the service ID and user name.
- Queue state – could be a single ID or a list of IDs, plus an index and a time offset. I think we need more than just a single ID, because even if the single ID is a playlist we'd want to resume playback from wherever the user left off.
- Target zone group.

There would be a complimentary web service that we'd expose to all content partners, the "Play-To Sonos Service". Our HHS talk to this service to associate Zone Groups with a particular user, and the content partner client talks to this service to enumerate Zone Groups and ask for a Play-To action to be initiated.

Poke holes, please..

-Ron

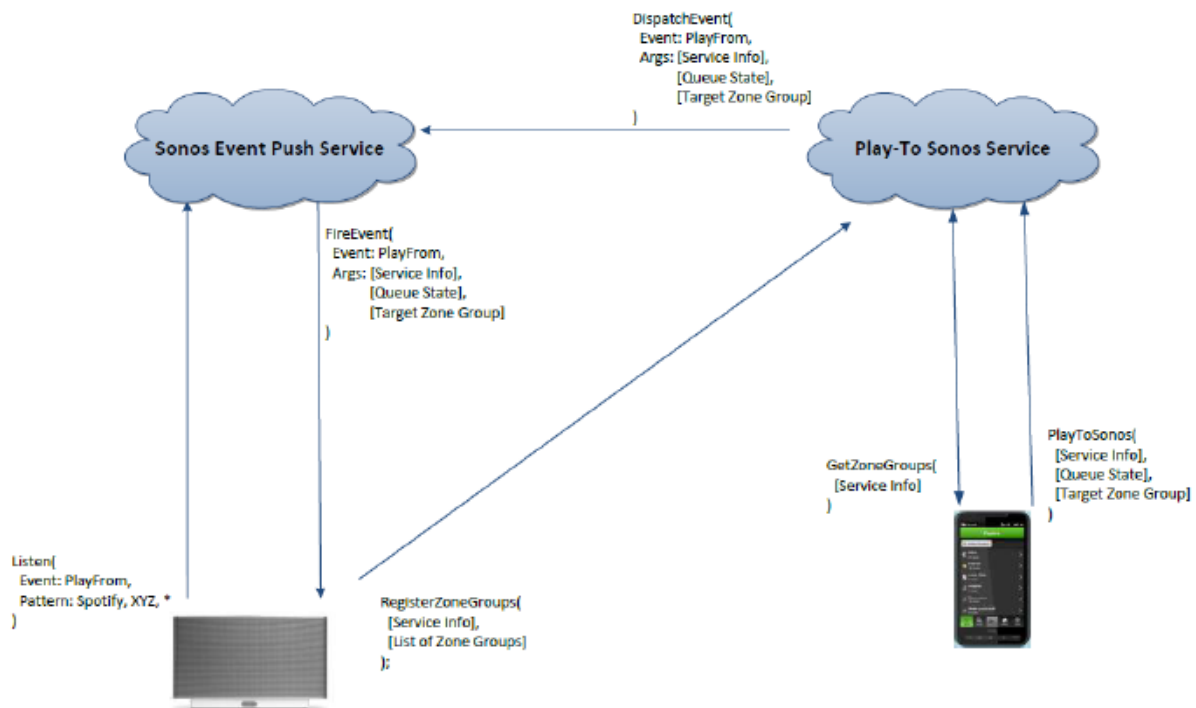
SONOS-SVG2-00027216. Mr. Kuper's email states that "one piece of this is that we want to introduce a new centralized web service for pushing events out to Sonos equipment." Ron Kuper explains that a Sonos speaker may connect to the service. The service will then "listen for events" and "push event" that "match a particular pattern" to the Sonos speaker. Mr. Kuper also explains that there will be a "Play-To-Sonos Service" that a third-party application can talk to "enumerate

Contains Highly Confidential AEO and Source Code Materials

Zone Groups and ask for a Play-To action to be initiated.”

105. The attachment to Mr. Kuper’s email provides a high-level illustration of the Play-to-Sonos feature Mr. Kuper described in his email. I have reproduced this illustration below and will now discuss the steps shows in the illustration:

Sonos Multi-Room Music Platform



SONOS-SVG2-00027229.

106. The illustration shows a “Sonos Event Push Service” and “Play-To-Sonos Service,” as well as a Sonos speaker and third-party application (e.g., Spotify).

107. In the example shown in the illustration, a Sonos speaker sends a message to the Sonos Event Push Service (reflected by the arrow from the speaker to the service) with a request to “Listen” for “PlayFrom” events. The Sonos speaker also sends a `RegisterZoneGroups` event to

Contains Highly Confidential AEO and Source Code Materials

the Play-to-Sonos Service to provide Service Info and Zone Group information. The third-party application receives the Service Info from the Play-to-Sonos service in a GetZoneGroups event.

108. Thereafter, the third-party application on the mobile device may dispatch a PlayToSonos event that takes in three parameters from the third-party application: (1) Service info; (2) Queue State; and (3) Target Zone Group. These parameters are described in Mr. Kuper's email above (SONOS-SVG2-00027216), for example the Queue State may be a "single ID or list of IDs" related to tracks in the third-party applications playback queue.

109. The PlayToSonos event is then relayed to the Sonos speaker. In particular, the PlayToSonos event is sent to the Play-To-Sonos service which relays it to the Sonos Event Push Service as a PlayFrom event. The Sonos Event Push Service then further relays the PlayFrom event to the Sonos speaker as part of a FireEvent. In other words, the Queue State sent by the third-party application is relayed to the Sonos speaker—it is *not* stored as a cloud queue in the Sonos cloud servers.

110. These documents do not disclose conception of the claimed invention. For example, to the extent Sonos argues that the mobile device running the third-party application is the "computing device" and that the PlaytoSonos event is the claimed instruction for transferring playback responsibility, there is no disclosure of the PlaytoSonos event configuring the Sonos speaker to "communicate with the cloud-based computing system in order to obtain data identifying a next one or more media items that are in the remote playback queue" and "use the obtained data to retrieve at least one media item in the remote playback queue from the cloud-based media service." The Speaker receives the FireEvent from the Sonos Event Push Service containing the PlaytoSonos event. The image does not illustrate any subsequent communications with the cloud service by the speaker. While the speaker is shown subscribing to a Sonos Event

Contains Highly Confidential AEO and Source Code Materials

Push Service to receive PlayFrom events through the Listen command, and is also shown sending the Play-to-Sonos Service a Register Zone Groups message, these are sent before the PlaytoSonos event is transmitted from the third-party application to the speaker. There is also no disclosure in this image that show the computing device “detect[s] an indication that playback responsibility for the remote playback queue has been successfully transferred,” let alone that the computing device transitions from the claimed first mode to the claimed second mode “after detecting the indication.”

111. These documents also confirms that Sonos had not conceived of a “remote playback queue” other than a playback queue in a third-party application. As the email explains, the third-party application provides the Queue State parameter, which may contain a single ID or a list of IDs, to the Sonos speaker.

112. **SONOS-SVG2-00027244:** Sonos also points to a July 14, 2011 email from Tad Coburn regarding Play to Sonos. In the email, Mr. Coburn is discussing the ability of “partners to integrate Sonos into their apps.” Mr. Coburn presents two approaches for doing so. In particular, in its interrogatory response Sonos appears to point to the discussion of a “throw a track (or other playable object) over the wall to Sonos” feature.” Mr. Coburn explains that this feature would involve the third-party application “passing the service-specific ID for a playable item (track, playlist, artist, programmed radio station, etc.) to Sonos and telling Sonos to either add the item to the [Sonos] queue or play it now.” A follow-up to this email explains “[t]here are two different possibilities for queue management. One is to give the app access to the Sonos queue. The other is for the app to override the Sonos queue with its own app-specific queue.” SONOS-SVG-00027082. In both scenarios, Sonos was considering queues stored locally on the Sonos device and queues in a third-party application.

Contains Highly Confidential AEO and Source Code Materials

playback system and the third party application to keep the local system and application synchronized." This disclosure refers to a local playback queue, a queue in a third-party application, and a shared queue-not a cloud queue. For the reasons I already explained, none of these provide adequate written description support for the term "remote playback queue" to the extent it is not limited to a queue in a third-party application.

710. The disclosure at 17:8-16 recites "[c]ertain embodiments facilitate streaming or otherwise providing music from a music-playing application (e.g., browser-based application, native music player, other multimedia application, and so on) to a multimedia content playback (e.g., Sonos™) system. Certain embodiments provide simple, easy-to-use and secure systems and methods for multimedia content playback across a plurality of systems and locations. Certain embodiments facilitate integration between content partners and a playback system as well as supporting maintenance of such content and system." Again, this disclosure merely refers to a user being able to edit and manage a music application, such as a third-party browser or mobile device application, in a third-party application, and then play back that queue on the local playback system. There is no discussion of a "remote playback queue" or "cloud queue."

711. Sonos also cites to Figures 6 and 7 of the patent. I already showed above that Figure 7 supports my opinion that the '033 patent does not provide written description support for the claims. Figure 6 (reproduced below) merely shows that a Zone Player can connect to the Internet or Local Area Network (LAN). The ability to connect to the Internet does not disclose a cloud queue. Indeed, the Sonos Zone Players that were available for purchase at the time of the alleged invention did not implement a cloud queue as I discuss in my section on written description, even though they could connect to an Internet or LAN. This disclosure at best teaches that the third-

Contains Highly Confidential AEO and Source Code Materials

713. Outside of the claims and Abstract (which was added after the initial written description was filed on December 30, 2011), the word "instruction" appears only four times in the written description. Each instance refers to processor instructions stored in memory. '033 patent at 3:3-7 ("Certain embodiments provide a computer readable storage medium including instructions for execution by a processor, the instructions, when executed, cause the processor..."), 8:11-21 ("processor 408 is a clock-driven electronic device that is configured to process input data according to instructions stored in memory 410... memory 410 is tangible machine readable medium storing instructions that can be executed by the processor 408."). These references do not disclose instructions that are transmitted by the computing device and are entirely irrelevant to this claim limitation.

714. The '033 patent discloses embodiments in which transferring playback from the third-party application to the local playback system causes the third-party application to pass a uniform resource indicator (e.g., a URL) to a proxy server, which then passes it to the local playback system. '033 patent at 12:41-64. The URI is then used to retrieve media content from a cloud server. *Id.* While the proxy server is in the cloud, there is no disclosure in the patent that the local playback system is configured to communicate with the cloud-based computing system after receiving the claimed instruction. Instead, the patent at most suggests that the one or more URLs are "pushed" to the local playback from the third-party application via the proxy server when transfer is initiated. *Id.* There is no disclosure of the local playback system receiving the claimed instruction and performing a "pull" in order to obtain data identifying a next one or more media items that are in the remote playback queue.

715. There is also no support for the full scope of this term as Sonos is interpreting it. In particular, for purposes of its infringement contentions, Sonos alleges that the full scope of sub-

Contains Highly Confidential AEO and Source Code Materials

elements (i) and (ii) encompass the alleged playback devices performing a series of three "pull" operations after receiving the alleged instruction. Namely, Sonos alleges that the accused Google cast receivers first perform a "pull" operation (namely, a GetWatchNext request and response) to obtain an identifier for a media item in a cloud queue; then perform a second pull operation to obtain a uniform resource locator (URL) from a cloud server (a Player Service) and then perform a third pull operation using the URL to obtain the media content. *See* 3-25-2022 Sonos Infringement Contentions, Ex. B ('033 Patent) at pp 69-73. The patent does not include any similar disclosure or otherwise provide support for the full scope of the claim under Sonos's reading.

716. I have reviewed Sonos's Validity Contentions as it relates to this claim term. *See* Validity Contentions at 264. I understand that Sonos identifies the following disclosures in the '033 patent as allegedly providing written description support for the term: 2:20-27, 2:61-3:23, 5:29-6:6, 7:7-11, 7:29-8:21, 8:31-55, 9:15-26, 9:54-62, 10:59-11:8, 11:9-12, 11:50-12:4, 12:6-14:15, 15:9-28, 15:34-53, 15:54-16:67, 17:1-16, FIGs. 6, 7. In my opinion, these disclosures do not provide written description support for the full scope of the claims, as I explain below. I note that Sonos's Validity Contentions do not meaningfully explain how Sonos contends these disclosures provide written description support for this limitation, and I reserve the right to respond to any explanation Sonos may provide in response to my opinions in this report. Moreover, as I already noted, Sonos's citations pick and choose from different embodiments in the specification that are never linked together.

717. The disclosure at 2:20-27 recites: "Certain embodiments facilitate streaming or otherwise providing music from a music-playing application (e.g., browser-based application, native music player, other multimedia application, and so on) to a multimedia content playback (e.g., Sonos™) system. Certain embodiments provide simple, easy-to-use and secure systems and

Contains Highly Confidential AEO and Source Code Materials

also PlaybackControls.java. The graphical interface displays the transport controls in a particular arrangement in both local and Tungsten playback modes (e.g., the skip to previous and skip to next buttons are arranged to the left and right (respectively) of the pause button).

3. Claim 25

784. I showed in my Opening '615 Report that the Tungsten/NexusQ prior art discloses or renders obvious claim 13. I incorporate the analysis in my Opening '615 Report by reference.

785. Claim 25 recites the additional limitations that the "control device" (i.e., the mobile device running the music application) include "a graphical interface," a "wireless communication interface to communicate with a playback device," and "one or more processors." As I showed in my Opening '615 Report, the Tungsten/NexusQ prior art includes "a graphical interface." *See* Opening '615 Rpt., e.g., ¶¶202-205. It also includes a "wireless communication interface to communicate with a playback device," such as a Wi-Fi interface. *See* Opening '615 Rpt., e.g., ¶¶206-228. Finally, I showed above in connection with Limitation 1.1 of the '033 patent that the Tungsten/NexusQ application runs on a mobile device that includes "one or more processors." Accordingly, in my opinion the Tungsten/NexusQ prior art discloses the additional limitations of Claim 25.

XVI. CLOUD QUEUE COLLABORATION

A. Sonos Has Accused Google Of Infringing Based On The Cloud Queue Technology That The Parties Developed As Part Of Their Collaboration

786. I discussed the Google Play Music ("GPM") application above in connection with the Tungsten/NexusQ prior art. As I explained, GPM is prior art to Sonos's '615 and '033 patents. The version of the GPM application (also referred to as the Music2 application) that is prior art allowed users to play back music on Google Tungsten/NexusQ devices, including cloud-hosted

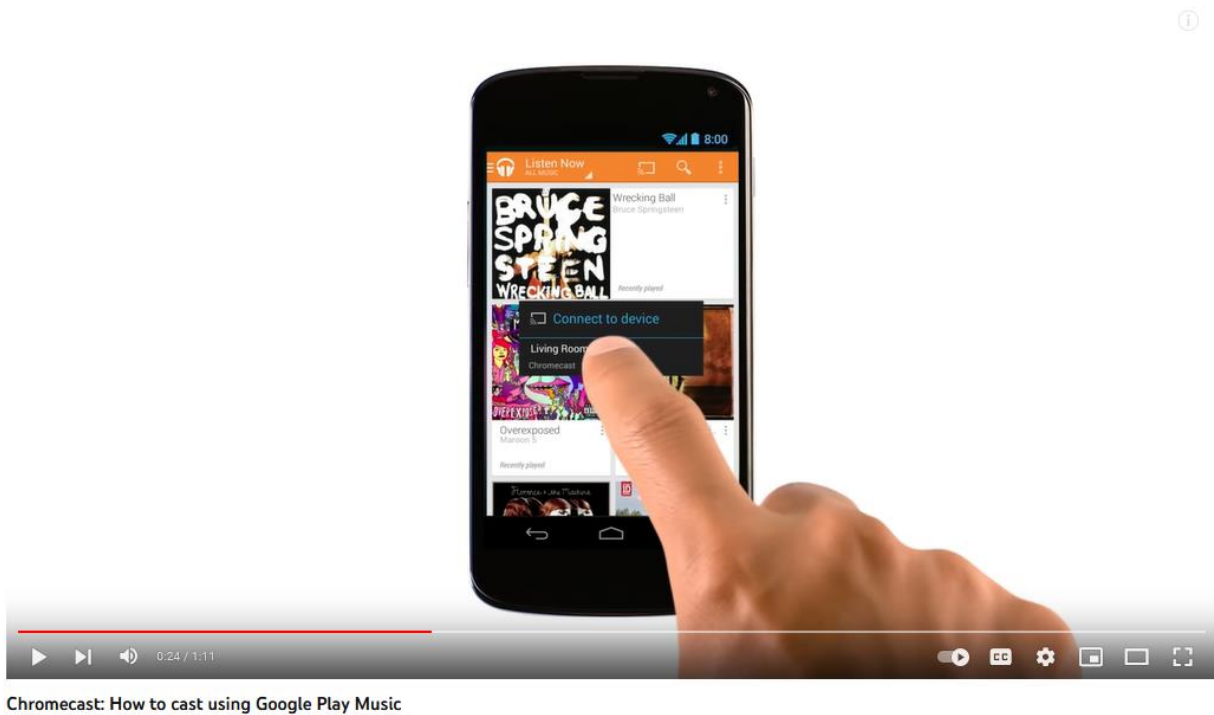
Contains Highly Confidential AEO and Source Code Materials

playlists (such as an album playlist or MagicPlaylist). A copy of the playlist was also stored on the playback device (the Tungsten/NexusQ device).

787. I understand that by 2013, Sonos and Google worked together to integrate Sonos's speakers with GPM. Dkt. No. 125 (Google's Second Amended Answer), 14-20. As part of this integration, Sonos and Google developed a Cloud Queue API that resulted in the playlist not being stored on the playback device—instead, a playback device would retrieve a window of tracks (e.g., the previous, current and next) from a cloud queue server that stored the playlist.

788. By the start of this collaboration, the GPM application also included the accused Cast features that Sonos is accusing, as seen in the following video: <https://www.youtube.com/watch?v=J2V0ITFzon4> (Chromecast: How to cast using Google Play Music, uploaded 11/27/2013); *see also* <https://web.archive.org/web/20150419091854/https://support.google.com/chromecast/answer/2998481?hl=en>. For instance, the image below from the video shows a user tapping the “Cast” icon to bring up available Chromecast devices that the GPM application could transfer playback to:

Contains Highly Confidential AEO and Source Code Materials

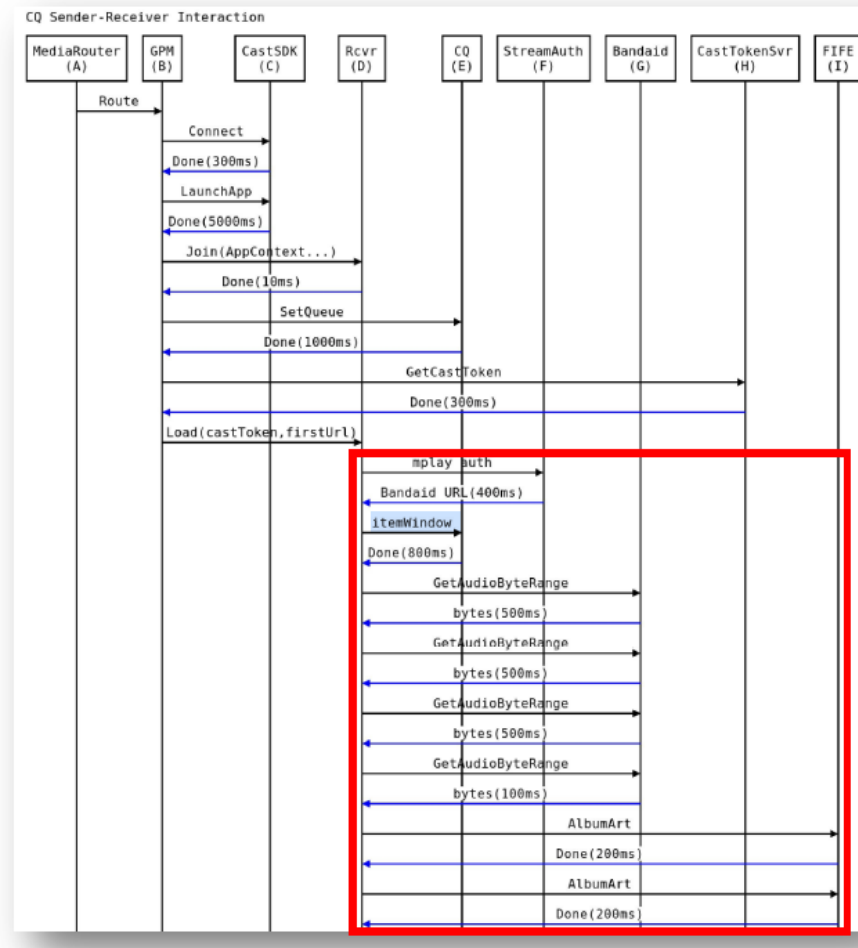


Id.

789. I understand that Google and Sonos executed several agreements in connection with their collaboration on the integration of Google Play Music. *Id.*, 21. For example, I understand that on November 13, 2013 Google and Sonos entered into a "Content Integration Agreement." Dkt. No. 125 (Google's Second Amended Answer), 21-22. I understand that Google is the "Service Provider" in the Content Integration Agreement and that Section 3.4 states that "Provider Developments"-which "consist of any and all development work done by or on behalf of Service Provider in creating the integrated Service offering"-and any and all intellectual property rights arising from or related thereto are and shall remain the sole and exclusive property of Service Provider":

Contains Highly Confidential AEO and Source Code Materials

792. For example, in its infringement contentions Sonos pointed to the below document as allegedly showing the accused Cloud Queue API functionality.

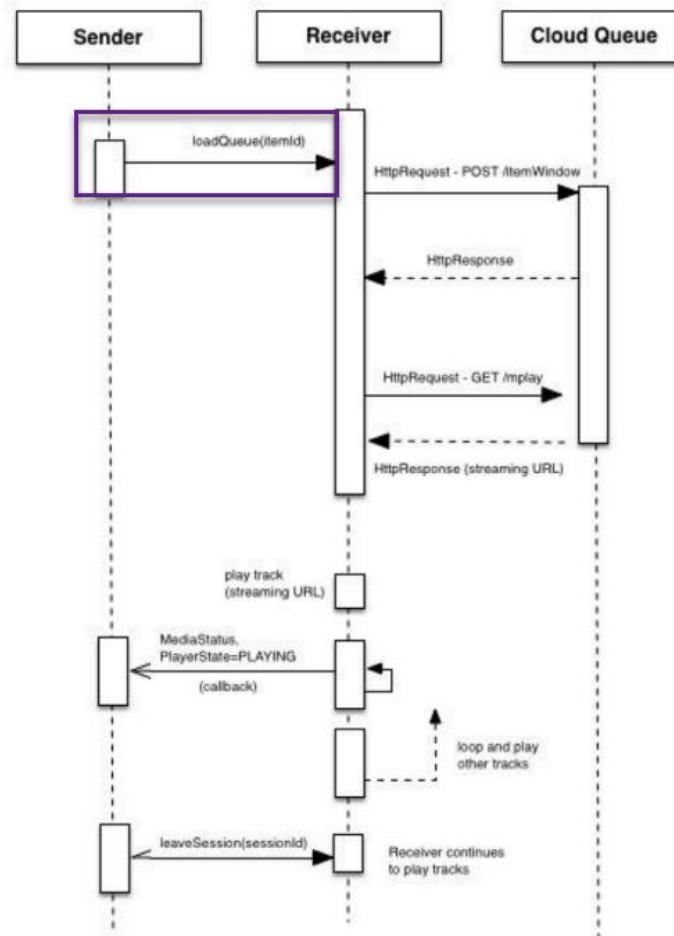


2-24-2022 Infringement Contentions, Ex. A ('615 Chart) at 94 (citing GOOG-SONOSWDTX-00043627). I also described the accused Cloud Queue API in my opening declaration in support of Google's Motion for Summary Judgment and Rebuttal '615 Report, which I incorporate by reference into this declaration.

793. The Cloud Queue API functionality that Sonos accused is the same functionality described in the Cloud Queue API documents from the parties' collaboration. For instance, just as shown in the image above of the accused Cloud Queue API, a sender device stores the playback

Contains Highly Confidential AEO and Source Code Materials

queue in the Cloud Queue and then issues a "load" command to start playback of the Cloud Queue on the receiver device. *See, e.g.*, "Music Cloud Queue JSON-based protocol for Commands and Events" (GOOG-SONOSWDTX-00040342) at GOOG-SONOSWDTX-00040360-62 (describing "loadCloudQueue" command). The load command is also shown in the Music Cloud Queue Sequence Diagrams that I have shown below (annotated in purple):

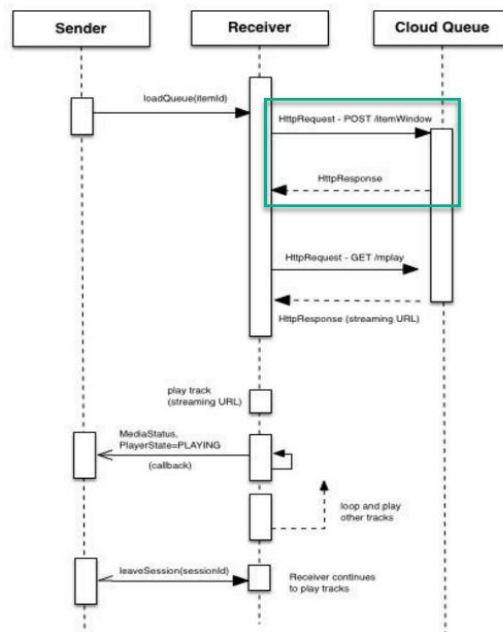


"Music Cloud Queue Sequence Diagrams" (GOOG-SONOSWDTX-00051100) at 1102; *see also* Music Cloud Queue JSON-based protocol for Commands and Events (GOOG-SONOSWDTX-00037316) at -7332-334 (describing loadCloudQueue command and "queueBaseURL").

Contains Highly Confidential AEO and Source Code Materials

794. Further, Sonos accused the ability of the Cloud Queue API in the accused GPM application to add ItemIDs and track URL (the alleged "resource locators") to a receiver device as part of an itemWindow response (the alleged "local playback queue"). *Id.* at 70-71. In particular, with the accused Cloud Queue API a receiver device transmits an itemWindow request to the Cloud Queue server specifying a "previousWindowSize" and "upcomingWindowSize." *See* MSJ Decl. (Dkt 210, Ex. 1), e.g., ¶112. The previousWindowSize and upcomingWindowSize are the number of tracks before and after the current media item that should be returned, which in the accused Cloud Queue API is set to one (1). In other words, in the accused devices the Cloud Queue will return an itemID and track URL for the current media item, as well as the one item before and after it. *Id.*

795. This same functionality is described in the Cloud Queue API documents exchanged during the collaboration. After receiving a "load" command, the Cloud Queue API involved sending an itemWindow request that returns an itemWindow response, as shown in the Music Cloud Queue Sequence Diagrams that I have reproduced below (annotated in green):



Contains Highly Confidential AEO and Source Code Materials

"Music Cloud Queue Sequence Diagrams" (GOOG-SONOSWDTX-00051100) at 1102.

796. Like the accused Cloud Queue API, the itemWindow request in the collaboration documents specified a "previousWindowSize" and "upcomingWindowSize." This can be seen in, for example, the Cloud Queue REST API which provides an example itemWindow response that has a "previousWindowSize" and "upcomingWindowSize" set to five (5) and twenty (20), meaning that the Cloud Queue server would return information for the current media item and the five media items before it and the twenty media items that after it. The "previousWindowSize" and "upcomingWindowSize" could also be set at one (1), as in the accused Cloud Queue API.

Getting the item window

GET <https://www.googleapis.com/musicqueue/v1.0/itemWindow?key=apikey>

Request contains the Cast Token in the Authorization header, the API key parameters as well as the following query parameters:

itemId (the sentinel value "" indicates the beginning of the queue)

previousWindowSize

upcomingWindowSize

[isExplicit](#)

Example:

GET <https://www.googleapis.com/musicqueue/v1.0/itemWindow?key=apikey&itemId=123456&previousWindowSize=5&upcomingWindowSize=20>

Response:

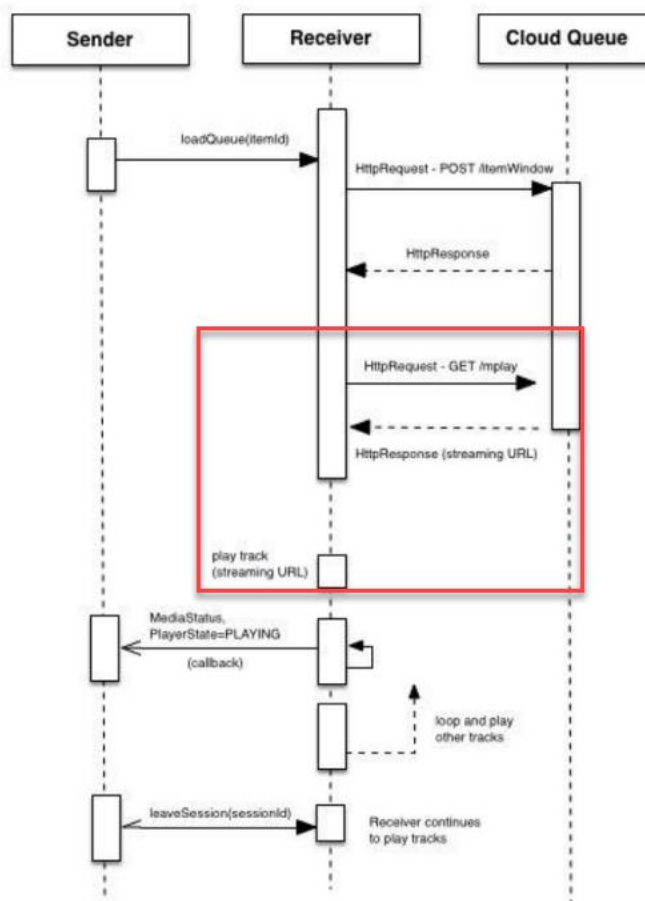
```
{
  "items": [
    {
      "kind": "musicqueue#itemReceiverData",
      "itemId": string,
      "trackUrl": string,
      "contentType": string,
      "deleted": boolean,
      "trackTitle": string,
      "albumArtist": string,
      "trackArtist": string,
      "albumTitle": string,
      "albumArtUrl": string,
      "durationMillis": long
    }
  ],
  "includesBeginningOfQueue": boolean,
  "includesEndOfQueue": boolean,
}
```

"Music Cloud Queue REST API" (GOOG-SONOSWDTX-00036998); see also "Debugging the Cloud Queue in five easy steps" (GOOG-SONOSWDTX-00037355) at 356-357.

797. Further, Sonos accused the ability of the Cloud Queue API in the accused GPM application to retrieve streaming URLs that it uses to obtain multimedia content from Google's content delivery network (called Bandid). *Id.* at 93. Again, this same functionality is described in the Cloud Queue API documents exchanged during the collaboration. For example, the Music Cloud Queue Sequence Diagrams that I have reproduced below shows the receiver obtaining a

Contains Highly Confidential AEO and Source Code Materials

streaming URL and then using the streaming URL to retrieve the multimedia content from the content delivery network (annotated in red):



"Music Cloud Queue Sequence Diagrams" (GOOG-SONOSWDTX-00051100) at 1102.

B. Sonos Shared The Cloud Queue Technology Developed During The Collaboration With Third Parties

798. I understand that after working with Google to develop the Cloud Queue API, Sonos prepared Cloud Queue API documentation that it shared with other companies.

799. For instance, SONOS-SVG2-00096259 is August 19, 2015 email in which Sonos employee Juergen Schmerder states "here is the Cloud Queue API doc for Spotify," and attaches a document entitled "Cloud Queue.pdf". I understand that Sonos engineer Keith Corbin, who was

Contains Highly Confidential AEO and Source Code Materials

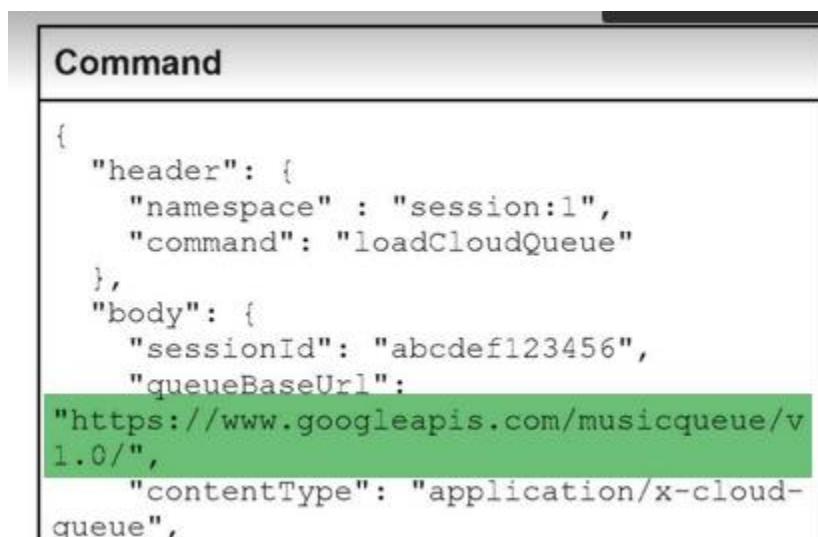
Cloud Queue Resource

The cloud queue resource is identified by a base URL:

<https://www.example.com/musicqueue/v1.0>

To tell Sonos which cloud queue to play, this URI must be passed to the `queueBaseUrl` parameter in the `loadCloudQueue` Control API action.

SONOS-SVG2-00096260. As I showed above, in the CloudQueue API the parties developed during the collaboration the GPM application also sent a “loadCloudQueue” command with a “queueBaseUrl” parameter to the playback device to tell it which cloud queue to play.” *See* Music Cloud Queue Sequence Diagrams” (GOOG-SONOSWDTX-00051100) at 1102; *see also* Music Cloud Queue JSON-based protocol for Commands and Events (GOOG-SONOSWDTX-00037316) at -7332-334. In fact, the base URL format in the Sonos Cloud Queue document is identical to the base URL format used in the cloud queue collaboration, except that Sonos has eliminated the reference to “Google” in the URL and replaced it with “example”:



Music Cloud Queue JSON-based protocol for Commands and Events (GOOG-SONOSWDTX-00037316) at -7332-334.

803. Similarly, the Sonos Cloud Queue document explains that after receiving the "load" command, the playback device sends an `itemWindow` request that returns an `itemWindow`

Contains Highly Confidential AEO and Source Code Materials

<p>Example</p> <pre> { "items": [{ "kind": "musicqueue#itemReceiverData", "itemId": "12345", "trackUrl": "http://example.com/path/12345.mp3", "contentType": "audio/mp3", "deleted": false, "trackTitle": "This is the track title", "trackArtist": "Some track artist", "albumArtist": "Album artist", "albumTitle": "Album title", "albumArtUrl": "http://example.com/path/foo.png", "durationMillis": 192000 }], "includesBeginningOfQueue": true, "includesEndOfQueue": true, "queueVersion": "version_string" } </pre>	<p>Response:</p> <pre> { "items": [{ "kind": "musicqueue#itemReceiverData", "itemId": string, "trackUrl": string, "contentType": string, "deleted": boolean, "trackTitle": string, "albumArtist": string, "trackArtist": string, "albumTitle": string, "albumArtUrl": string, "durationMillis": long }], "includesBeginningOfQueue": boolean, "includesEndOfQueue": boolean, "queueVersion": string } </pre>
--	---

SONOS-SVG2-00096268-269 (left); GOOG-SONOSWDTX-00036998 (right).

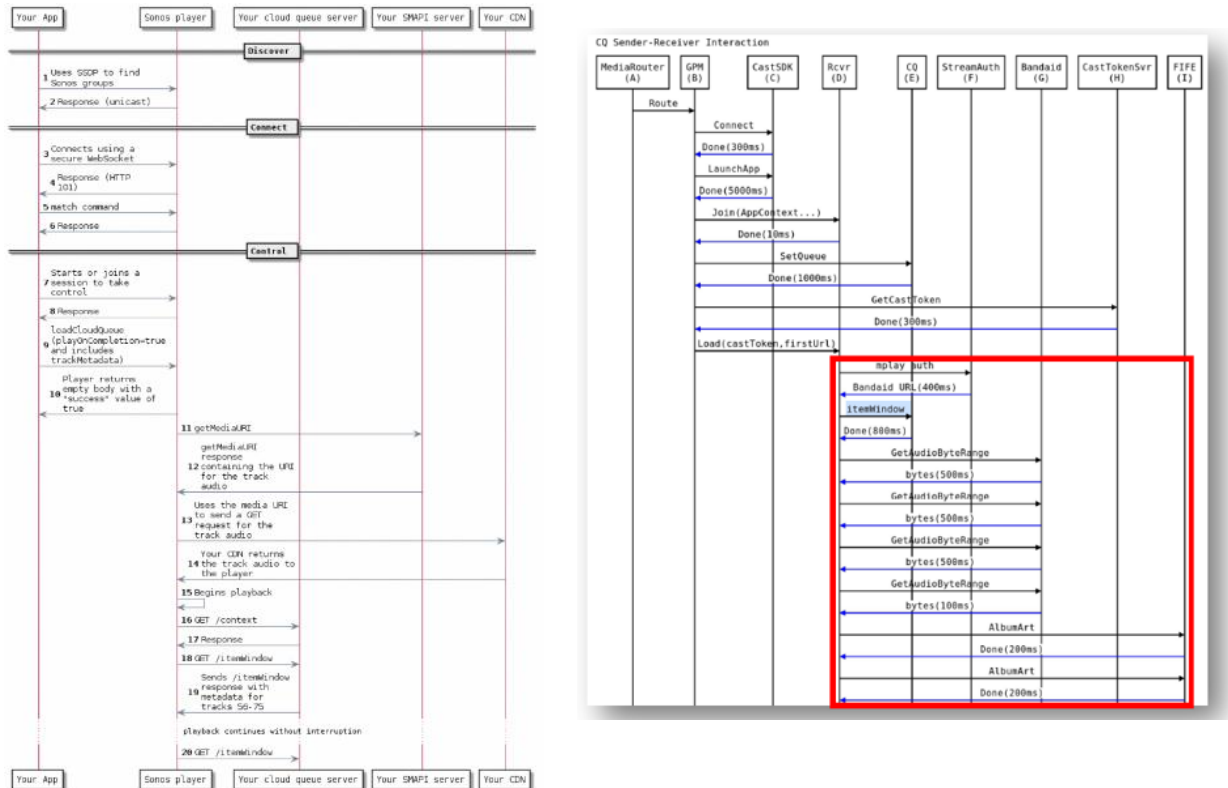
804. As can be seen in the above image, the itemWindow response includes a “trackURL.” The trackURL is described as the “URL for the track.” SONOS-SVG2-00096268-271. Thus, just like the Cloud Queue API developed during the collaboration, the playback device receives URLs that are used to retrieve media content.

805. Similarly, I have reviewed Sonos’s developer website and its discussion of Sonos current “Cloud Queue API” version. <https://developer.sonos.com/reference/cloud-queue-api/> (“The current version number of the cloud queue API is v2.3”). In my opinion, the Cloud Queue API that is described on Sonos’s developer website is based on the Cloud Queue API that the parties developed during the collaboration, although Sonos appears to have added some additional commands and features in later versions of its Cloud Queue API.

806. For example, Sonos’s developer website explains that “[o]ne way to play audio on Sonos is by using a cloud queue, a list of tracks that you host on a server that the player can access.” <https://developer.sonos.com/reference/control-api/playbacksession/load-cloud-queue/>. The

Contains Highly Confidential AEO and Source Code Materials

Sonos Developer website includes a “Cloud queue workflow,” which is based on the Cloud Queue API the parties developed and that Sonos accused.



<https://developer.sonos.com/build/content-service-get-started/play-audio/>.

807. For example, the Sonos developer Website instructs developers to “[u]se the loadCloudQueue command in the playbackSession namespace to load, and optionally start playback of, an item in a cloud queue.” *Id.* The loadCloudQueue command described on the Sonos website includes, for instance, a queueBaseURL for the cloud queue, an ItemId for the track, a positionMillis parameter that identifies the position within the track that playback should start at milliseconds, track metadata, and a URL for the track. The loadCloudQueue command developed during the parties’ collaboration also included this information, as can be seen in the following

Contains Highly Confidential AEO and Source Code Materials

image which shows an example loadCloudQueue request from the Sonos developer website (left) and an example from the collaboration documents:

Example	
<p>Request</p> <pre> 1 { 2 "queueBaseUrl": "https://www.example.com/musicqueue/1243328192349892/v2.0/", 3 "httpAuthorization": "Bearer 1tg1343f-xyz9-300t-5380-8y4545y2p400", 4 "itemId": "abc", 5 "queueVersion": "asdf3cbjal235jozz", 6 "positionMillis": 30000, 7 "playOnCompletion": false, 8 "trackMetadata": { 9 "album": { 10 "name": "Album Name" 11 }, 12 "name": "Track Name", 13 "artist": { 14 "name": "Artist Name" 15 }, 16 "imageUrl": "http://example.com/path_to_image_url", 17 "durationMillis": 319000, 18 "contentType": "audio/mpeg", 19 "mediaUrl": "http://example.com/path_to_track_url" 20 } 21 }</pre>	<pre> "queueBaseUrl": "https://www.googleapis.com/musicqueue/v1 .0/", "contentType": "application/x-cloud-queue", "httpHeaders": { "Authorization": "playon=myCastToken" }, "itemId": "abc", "positionMillis": 0, "playOnCompletion": false, "firstTrackMetadata": { "trackTitle": "Foo", "trackArtist": "Bar", "albumArtist": "Bar", "albumTitle": "Baz", "albumArtUrl": "<u>https://path.to.album/art</u>", "durationMillis": 180000, "trackUrl": "https://android.clients.google.com/music /mplay?songid=foo", "contentType": "audio/mpeg" } }</pre>

<https://developer.sonos.com/reference/control-api/playbacksession/load-cloud-queue/> (left);

Music Cloud Queue JSON-based protocol for Commands and Events (GOOG-SONOSWDTX-00037316) at -7332-334 (right).

808. The Sonos developer website also discloses that, just like the Cloud Queue API developed by the parties, when the loadCloudQueue command is received the playback device retrieves a URL to begin playing the first track, and also retrieves a window of tracks from the cloud queue through an itemWindow request and response.

809. For example, the Sonos developer website explains that the itemWindow request includes the parameters “isExplicit,” “itemId,” “previousWindowSize,” “upcomingWindowSize,” and “queueVersion,” which are the same parameters in the itemWindow request that the parties developed during the collaboration. Sonos also added an additional “reason” parameter in Version 2.2:

Contains Highly Confidential AEO and Source Code Materials

Request

The request contains the access token in the authorization header and the parameters in the table below.

Parameter	Type	Description
<code>isExplicit</code>	boolean	If true, indicates that a user performed a specific action or there was a specific intent to play the track. For example, the user pressed play, skipped backwards or forwards, or selected the track in the queue. If false or omitted, indicates that the player requested the track, for example, because it was the next track in the queue. You can use this to track user actions, for example, to determine if the playback stream has been initiated from another device.
<code>itemId</code>	string	The identifier for the track that should be returned. If it is omitted or an empty string, it indicates your server should return the first item in your cloud queue.
<code>previousWindowSize</code>	number	The maximum number of tracks before the track specified by the <code>itemId</code> that should be returned by your cloud queue server. A non-negative integer.
<code>upcomingWindowSize</code>	number	The maximum number of tracks after the track specified by the <code>itemId</code> that should be returned by your cloud queue server. A non-negative integer.
<code>queueVersion</code>	string	<p>The last cloud queue change state identifier cached by the player. This could have been from:</p> <ul style="list-style-type: none"> the last <code>GET /itemwindow</code> or <code>GET /version</code> response. a <code>loadCloudQueue</code> or <code>skipToItem</code> response. <p>This is omitted when the value is unknown, for example, if the server did not respond to a query.</p>
<code>reason</code>	string	<p>The reason a player is requesting a new item window. See below for a list of possible values.</p> <p>New in v2.2: Players can send requests with multiple <code>reason</code> values separated with a + symbol. For example, <code>reason=load+queueCompleted</code>.</p>

<https://developer.sonos.com/reference/cloud-queue-api/get-itemwindow/>.

810. Similarly, the Sonos developer website explains that the `itemWindow` response includes the parameters “items,” “includesBeginningofQueue,” “includesEndofQueue,” and “queueVersion,” which are also parameters that were included in the Cloud Queue API developed by the parties. Sonos also added an additional “contextversion” parameter in Version 2.0:

Contains Highly Confidential AEO and Source Code Materials

<https://developer.sonos.com/reference/cloud-queue-api/get-itemwindow/>. And like the Cloud Queue API, the Sonos developer website explains that the playback device receives URLs that are used to retrieve media content from a Content Delivery Network.

C. Sonos Filed A Patent On The Cloud Queue API

811. I understand that on June 3, 2014, while Sonos and Google were collaborating on the cloud queue development, Sonos filed U.S. Provisional Application No. 62/007,906 (“the ’906 Provisional”) entitled “Cloud Queue,” which names Steven Beckhard and Arthur L. Coburn, IV as inventors. The ’906 Provisional does not name any Google employees as inventors on its face. AKERMAN-2497. I understand that Sonos has also obtained a number of patents which claim priority to the ’906 Provisional, by way of example U.S. Patent No. 9,654,459.

812. I understand that Sonos employee, Steven Beckhardt, testified that the ’906 Provisional was a patent on the cloud queue API. Beckhardt Tr. at 93:8-14 (“Q. Earlier you had testified that you had filed a patent on the cloud queue API. Do you recall that? A. Yes. Q. Is this the patent that you were referring to? A. I think so.”). Indeed, the ’906 Provisional includes a section entitled “Queues in the Cloud.” AKERMAN-2513. The terms “Queues in the Cloud” is used in documents from the time of the collaboration to refer to the Cloud Queue API that the parties were collaborating on. For example, in a 12-11-2013 email from Tad Coburn with the subject line “P2S [Play-To-Sonos] goals for 2014,” Mr. Coburn states “[c]ontinue to work with Google post-Biggie to design and implement a solution that passes the ‘beer test’ (a.k.a. ‘queues in the cloud’”). SONOS-SVG2-00073496 (12-11-2013 email from Tad Coburn); SONOS-SVG2-00080156 (5-9-2014 email from Debajit Ghosh to Tad Coburn with the subject “Re: ‘queues in the cloud’ status?”); *see also* Beckhardt Tr. at 95:20-25 (“Q. Do you recall seeing today documents that showed the cloud queue API that Google and Sonos were working on was referred to as a queues in the cloud? A. Yes.”); *see also id.* at 98:3-7 (“Q. Were you aware of any other cloud

Contains Highly Confidential AEO and Source Code Materials

queue APIs that Sonos was working on in 2014 other than with Google? A. Sorry, I don't recall.”). Moreover, a substantial portion of the written description in the '906 Provisional application describing the “Queues in the Cloud” appears to have been copied from the Queues in the Cloud documents I discussed in my section on the cloud queue collaboration. *Compare* SONOS-SVG2-00233699 (Queues in the Cloud, Version 44) *with* AKERMAN-2513-521. For example, the '906 Provisional includes the same basic description of Playing a CloudQueue that I discussed above, including that the client application sends to the playback device a URL to a cloud queue and that the playback device then fetches a window of tracks from the cloud queue:

Playing a CloudQueue

Once a Sonos player has the URI to a CloudQueue (or Playhead), it can "play" that cloud queue. This works similarly to a SMAPI programmed radio station, but with a more sophisticated API that allows bi-directional events and updates.

Transport control commands (Play, Pause, Skip Next, Skip Previous, Seek to Track, Scrub to Position within Track) can be sent either directly from the control app to the Sonos player across the LAN (which is optimal) or from the control app to the PlayHead and then down to the Sonos player. The control app should receive transport-related events through the same path that is used to send control commands.

Basic Playback

1. The player fetches an initial window of tracks from the CloudQueue and stores (caches) it locally. This window should be at least two tracks to allow the player to perform gapless playback.
2. The player fetches the first track in the window and begins to play it.
 1. What are the authentication mechanism(s) for each track? SMAPI account (yes), OAuth token (what happens if it expires?), custom HTTP header (what happens if the token in the header expires?)
3. When the player advances to the last track (or possibly a track or two before that), if it believes there are more tracks in the CloudQueue, it fetches the next window of tracks from the cloud and stores (caches) it locally.
4. Each time the player advances to the next track (or in general, jumps to a different track), if the player is connected to a Playhead in the cloud, the player sends an event to the Playhead to indicate the old track that just finished playing (and perhaps the position within that track, so the cloud knows whether it was played through to the end?) and the new track that has started playing.
5. When the player reaches the end of the entire list of tracks in the CloudQueue, it moves on to the first container in the "Future" list (assuming there are no special modes in effect).
 1. See below for special modes that can affect what the Sonos player does when it reaches the end of a CloudQueue.

Akerman-2518.

Contains Highly Confidential AEO and Source Code Materials

XVII. INVALIDITY UNDER SECTION 102(F)

813. I showed above that Sonos interpreted the '615 patent to accuse Google Play Music of infringing based on the Cloud Queue API that the parties developed during the collaboration. *See* Section XVI [Cloud Queue Collaboration].

814. I understand that in August of 2020, prior to the issuance of the '033 patent, Google announced that it was replacing Google Play Music with YouTube Music. <https://blog.youtube/news-and-events/youtube-music-will-replace-google-play-music-end-2020/>. Thus, while Sonos is not accusing Google Play Music of infringing the '033 patent, Sonos is interpreting the Asserted Claims of the '033 patent to accuse cloud queue technology in the YouTube applications that is in relevant respects the same as the Cloud Queue API that the parties were developing during their collaboration.¹⁴⁰

815. For instance, Sonos alleges that a `setPlaylist` message is sent to a Cast-enabled receiver that causes it to playback a “remote playback queue,” which Sonos is interpreting to encompass a cloud queue. *See, e.g.*, 11-21-2022 Sonos Infringement Contentions at 43-45. Thus, the `setPlaylist` message is similar to the `loadCloudQueue` command in the Cloud Queue API. Sonos further alleges that after receiving a `setPlaylist` message a Cast-enabled receiver will send a `WatchNext` request message to obtain a window of tracks from the cloud queue (e.g., the previous, current, and next items in the cloud queue). *Id.* Thus, the `WatchNext` message is similar to the `itemWindow` request in the Cloud Queue API. Additionally, I understand that Sonos alleges that

¹⁴⁰ Indeed, I understand that Sonos’s counsel initially characterized the '033 patent as one of its “cloud queue patent” that “deal with exactly how music streaming on a cloud gets transferred between a device, let’s say a cell phone, and a speaker and how do you take a queue that’s queued up in the cloud and transfer that queue of music from one device to another.” *See* Nov. 11, 2020 Hr’g Tr. 17:3-8. I understand that Sonos now claims that its counsel was mistaken in referring to the '033 patent as a cloud queue patent, although it continues to accuse Google’s cloud queue technology.

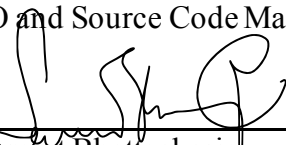
Contains Highly Confidential AEO and Source Code Materials

a Cast-enabled receiver will use the information in the WatchNext response to retrieve a media item by making a request to a Player Service for a Bandid URL and a subsequent request to Bandid for the content for playback. *Id.* In the CloudQueue API, a Cast-enabled receiver similarly made a mplay URL request to obtain a Bandid URL and then a request to the Bandid URL to obtain the content for playback.

816. I disagree that the accused cloud queue technology infringes the Asserted Claims of the '033 patent. It is also my opinion that the Asserted Claims are invalid, including because they are not novel and non-obvious over the prior art. Indeed, I understand that the Court has already found that the '615 patent is invalid over the prior art, and the claims of the '033 patent are substantially the same-with the primary difference being that the '033 patent receives a "remote playback queue," while the '615 patent recites a "local playback queue."

817. Nevertheless, to the extent Sonos's infringement and validity allegations are credited, then it is my opinion that the Asserted Claims are invalid for improper inventorship. The persons named on the '033 patent as inventors (Tad Coburn and Joni Hoadley) did not, by themselves, conceive of every feature of the subject matter sought to be patented. For instance, based on my review of the documents and testimony in this case, it is my opinion that the cloud queue technology that was developed during the parties' collaboration was a result of a collaborative process between Sonos and Google, and, at minimum, both Google and Sonos were jointly responsible for conception of the cloud queue technology. In other words, to the extent that Sonos asserts that the '033 patent is supposedly infringed by the accused cloud queue technology, then the '033 patent is invalid for failure to correctly join the Google employees who collaborated with Sonos to conceive and reduce to practice the cloud queue technology, for example Debajit Ghosh.

Contains Highly Confidential AEO and Source Code Materials



Samrat Bhattacharjee